2024 Spring Artificial Intelligence
Final Project-Large Language Model

Team 12 蔡芳慈 111550024 陳奕 111550168 謝詠晴 111550113 郭芷杼 111550155

Topic:Exploring Text-Emoji Correspondence

Introduction

In the rapidly evolving digital communication landscape, emojis have become an integral part of how we convey emotions, reactions, and context in text-based conversations. However, selecting the most appropriate emoji for a given context can be time-consuming. We need to switch to the emoji keyboard first, select the emoji, and then switch back. Through this emoji prediction system, we can simplify digital communication by suggesting suitable emojis, thereby saving users time and effort. It enhances clarity and understanding of text-based conversations by providing additional contextual and emotional cues. Additionally, it promotes inclusivity by facilitating cross-cultural communication through universally understood visual symbols. In our work, we intend to train a Chinese text-to-emoji prediction system and analyze different performance of our model using different numbers of classes and models.

Literature Review / Related work

Prior works used the distillbert-based-uncased model to predict, and there was no other data processing except tokenization. They have a similar input format which are in English, and some of them have an output format containing fixed multiple emojis(lyrics to emoji : https://www.kaggle.com/code/aguschin/lyrics-to-emoji), and some of them allow the user to specify the number of output emojis(text to emoji : https://github.com/andylolu2/Text2Emoji). In our work, the input is in Chinese, and it will output only one emoji for a single sentence.

Dataset

由於現成的資料集無法滿足我們對包含特定表情符號的中文資料的需求，我們嘗試過翻譯related work的英文資料集，但翻譯效果不理想，並且存在一定的文化差異。因此，我們決定自行爬取資料。考慮到某些 API 需要付費，最終我們決定使用經常使用表情符號的 Instagram 作為資料來源。

數據收集：我們從 Instagram 的 22 個帳號中收集數據，每個帳號收集 2000 筆資料。對於每篇貼文，我們提取了文章內容以及20條包含表情符號的留言。選擇這些帳號的原因為：

1. 包含不同類型的帳號及對應留言(名人帳號、論壇帳號...)
2. 這些帳號使用到大量emoji

以下是我們選擇的帳號：

foodiefoodiego、dcard.tw、megaportfest、7elevente、william_chingte、jessetang11、netflixtw、_lonely_foodie_、museacg、ashin_ig、jaychou、104student、girlstyle.mag、yi.maooo、imqingfeng、__tingjia.zhang__、

mcdonaldstw、mcdonaldstw、truedantw、yucheno3o1、yoshihikoshindo、emily___life___、toyotatw

數據預處理：將收集到的數據整理並保存為 CSV 檔案。並進行以下預處理

1. remove user names（ex, @jaychou）
2. cut the words
   使用jieba的繁體字典庫dict.txt.big，進行繁體中文分詞及處理。
3. remove stopwords & punctuations
   標點符號和停用詞（例如「的」、「了」、「在」等）在文本分析中通常是無用的，因此需要被移除。我們可以從一個預先定義的停用詞檔案stopwords.txt中讀取停用詞。
4. remove non-chinese words

以上操作程式碼檔案為text_processor.py。

在預處理過程中，我們去除了只包含表情符號的留言，以及經過預處理後變為空的資料。確保我們分析的數據具有實質性的文本內容。

表情符號分析：我們統計了最常用的前 20、30、40個表情符號，並保留了包含這些表情符號的資料數據，程式碼檔案為"emoji_statistic_and_filter.py”。

數據統計：

- 總共收集的數據：最初，我們總共收集了 41273 筆資料
- 預處理後的數據：在去除只包含表情符號和空留言後，我們剩下 35507 筆有效資料。
- 最終數據集：在篩選出包含前 20、30、40 個最常用表情符號的資料後，我們最終得到 20078、22783、24596 筆符合條件的資料。

Baseline

We are using a baseline model that takes translated text as a dataset. In this simplified version, the model is designed to output only the 20 most commonly used emojis. For this task, we utilize a training dataset obtained from the web via CodaLab*. The dataset contains 50,000 english tweets from X (formerly known as Twitter), each line contains one or more emojis. We translate all the tweets through Google Translate into Chinese as our dataset.

To train our model, we employ a Bidirectional Long Short-Term Memory (BLSTM) neural network. BLSTM is a powerful variant of the standard LSTM that can process data forward and backward, allowing it to capture the context of past and future states, which is particularly beneficial for understanding nuances in text and predicting appropriate emojis.

The training process involves feeding the BLSTM model tweets from the dataset and iterating over multiple epochs to optimize its performance. The goal is to enable the model to effectively learn the relationship between text and corresponding emojis.

Once trained, the model will be able to take new Chinese text as input and accurately predict the most appropriate emoji from a predefined set of 20 commonly used emojis.

In baseline, the model is limited by translated data. Since the content translated by Google Translate is not close to the way Taiwanese speak, and the emoji commonly used in English tweets are different from those of Taiwanese, the output results may be unnatural. Based on this, we decided to train the model using the data we collected as the training dataset.

*https://competitions.codalab.org/competitions/17344#learn_the_details-overview

Main Approach

我們使用雙向長短期記憶網絡(BLSTM)來預測中文文本中的表情符號。BLSTM是一種先進的循環神經網絡(RNN)，能夠在正向和反向兩個方向上處理輸入序列，從而捕捉到文本的上下文，這對於理解文本的語義非常重要。
以下是我們的訓練流程:

1. Data Preprocessing
    a. 如上述Dataset部分所述。
    b. Over-sampling Data
    由於每個emoji的數據量不相符，我們利用RandomOverSampler解決數據不平衡問題，這樣可以確保模型在訓練過程中不會偏向數據量較多的表情符號。
    c. Tokenize
    利用tokenizer將dataset中的單詞以詞頻排序後編號，再將每個文本都轉化成以編號表示的序列。這樣的處理可以使模型更有效地理解文本中的詞語。
    d. Padding
    為了確保每個文本的長度一致，我們對文本進行填充或截斷，使所有文本的長度相同。
    以文本「白桃烏龍茶雪華夫餅 整份$204、半份$116」為例:
    1. 經過dataset提及的數據預處理後，轉化為['白桃', '烏龍茶', '雪華夫餅', '整份', '半份']
    2. 經過tokenize後變為[3536, 3980, 3981, 1854, 1855]
    3. 經過padding後，轉化為:

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0
                ……
   0   0   0   0   0   0   0   3536  3980  3981  1854 1855 ]
```

2. Model Architecture
    a. 嵌入層
    將input轉換為固定大小的密集向量。這些向量可以捕捉詞語之間的語義關係，為後續的BLSTM層提供有效的特徵表示。

b. BLSTM層
利用兩個BLSTM層捕捉上下文。

c. 全連接層
將BLSTM輸出映射到所需的輸出大小，即表情符號的機率分布。其中activation function為Softmax，確保輸出的機率總合為1，以便最終的表情符號預測。

參數設定詳見下圖:

```
vocabulary_size = vocab_size
seq_len = 140
embed_len=128


model = Sequential()
model.add(Embedding(vocabulary_size+1, embed_len, input_length=seq_len))
model.add(Bidirectional(LSTM(80,return_sequences=True)))
model.add(Bidirectional(LSTM(80,return_sequences=True)))
model.add(GlobalMaxPool1D())
model.add(Dropout(0.5))
model.add(Dense(64, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(20, activation="softmax"))
```

1. 在data preprocessing後的結果作為input:
以文本「白桃烏龍茶雪華夫餅 整份$204、半份$116」為例:

[0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
……
0 0 0 0 0 0 0 3536 3980 3981 1854 1855]

2. 最終的輸出為每個表情符號分別的機率
例如: [0.15, 0.01, 0.03, …, 0.02]

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 140, 128)          2317056

 bidirectional (Bidirection  (None, 140, 160)          133760
 al)

 bidirectional_1 (Bidirecti  (None, 140, 160)          154240
 onal)

 global_max_pooling1d (Glob  (None, 160)               0
 alMaxPooling1D)

 dropout (Dropout)           (None, 160)               0

 dense (Dense)               (None, 64)                10304

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 20)                1300

=================================================================
Total params: 2616660 (9.98 MB)
Trainable params: 2616660 (9.98 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

3. Training Process

a. Loss function
利用Categorical Cross-Entropy來測量預測準確性。

b. Optimizer
利用Adam optimizer進行高效的梯度下降。Adam是一種自適應學習率的優化方法，能夠在訓練過程中自動調整學習率，從而加快收斂速度和提高模型性能。

c. Batch Training
利用batch training優化權重，其中batch size為64、epoch為15，並使用early stopping。

4. Prediction

a. Preprocessing
如1.所述，對新輸入的文本進行處理。

b. Model Inference
將已預處理的input文本傳入BLSTM模型。

c. Output
從Softmax輸出中選擇一率最高的emoji。

以文本「大家好我是露營系大二的阿智，因為太常辦營隊被爸媽說我讀露營系。平常的興趣是睡覺打羽球、聽deca joins還有吃好吃的抹茶。最近在練習看動漫，大家可以推薦窩好看的動漫。常常被說很兇，一定是誤會。常常被阿林叫去洗碗，但是還沒洗過。在這裡要謝謝我的爸爸，謝謝他花1120元買了八個NCTU CS系枕頭送給其他大學長，這個週末要去參加他們的同學會，希望他們可以順便賞我一個工作。大家可以來資工之夜看我跟 阿皓 啊源 小黃 小李表演光球！期待辦資工營噎」為例，經預測的output*如下：

```
1/1 [==============================] - 0s 156ms/step
1/1 [==============================] - 0s 152ms/step
1/1 [==============================] - 0s 141ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 157ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 150ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 151ms/step
1/1 [==============================] - 0s 139ms/step
1/1 [==============================] - 0s 135ms/step
1/1 [==============================] - 0s 131ms/step
1/1 [==============================] - 0s 143ms/step
1/1 [==============================] - 0s 136ms/step

大家好我是露營系大二的阿智👍
因為太常辦營隊被爸媽說我讀露營系🥹
平常的興趣是睡覺打羽球、聽deca joins還有吃好吃的抹茶👉
最近在練習看動漫😂
大家可以推薦窩好看的動漫❤
常常被說很兇✨
一定是誤會😂
常常被阿林叫去洗碗😂
但是還沒洗過🔥
在這裡要謝謝我的爸□□
謝謝他花1120元買了八個NCTU CS系枕頭送給其他大學長🥰
這個週末要去參加他們的同學會👍
希望他們可以順便賞我一個工作✨
大家可以來資工之夜看我跟 阿皓 啊源 小黃 小李表演光球！期待辦資工營噎🔥
```

5. UI

我們利用flask開發了一個網站，串接上述的模型，使用者可以輸入一段文字，網站將以換行做為每句話的分隔，為每句話加入適當的表情符號。

以輸入文本

「我是真的不能控制我自己

每天想你 甚至會出現在夢裡
把對你的情緒全部哼成旋律
還是期待幻想的畫面會不會發生」
為例，會得到以下結果：

*部分emoji無法被電腦顯示，將以🥹表示。

Evaluation Metric

定量指標

1. f1-score of each class(emojis)

$$f1\ score = \frac{2 * Precision * Recall}{Precision + Recall}$$
$$Precision = \frac{TP}{TP+FP}$$
$$Recall = \frac{TP}{TP+FN}$$

2. accuracy of total prediction

$$accuracy = \frac{f1\ score\ of\ each\ emoji * support}{total\ number\ of\ supports}$$

3. confusion matrix of each class
   for an n*n confusion matrix C, i means each index of emoji

$$True\ Positive_i(TP) = C[i,i]$$
$$False\ Positive_i(FP) = \sum_{k=0} C[k,i] - C[i,i]$$
$$False\ Negative_i(FN) = \sum_{k=0} C[i,k] - C[i,i]$$

$$True\ Negative_i(TN) = \sum_{k=0}\sum_{q=0} C[k,q] - TP_i - FP_i - FN_i$$

定性指標

1. 易用性:UI與操作流程是否方便
2. 可解釋性:使用者是否可以知道表情符號與文字的關係
3. 用戶體驗:增加表情符號是否幫助文章理解

Results & Analysis

Results

1. 翻譯Baseline的dataset, 測試中文是否能成功

   我們先採用baseline所使用的twiter datasets, 取3000筆進行train, 並利用google sheet的"=GOOGLETRANSLATE(A2,"en","zh-TW")" 函式將原先英文的資料翻譯成中文。

```
[ ] df.head(10)
```

	Tweet	Label
0	小復古最愛的人水牆	0
1	華麗昨天 kcon 化妝 羽毛	7
2	民主廣場喚醒令人震驚的結果決定 NBC 新聞	11
3	amp vilo 華特迪士尼魔法王國	0
4	銀河系很遠很遠	2
5	今晚佛羅裡達晚餐 煎鮭魚 蒸粗麥粉 蔬菜沙拉 美味的晚餐 佛羅裡達鮭魚	1
6	最喜歡的高級比賽恭喜擊敗西西塞勒姆	8
7	得到了正式的最好的朋友 phi mu jsu	0
8	原因想念小兄弟復古表弟愛印第安納大學	13
9	生日吻麥迪遜威斯康辛州	9

翻譯後的dataset會再經過jieba套件斷詞並移除stopwords。

- idx2words = tokenizer.index_word 後的結果

```
Word Mappings

Index:  1 ----> 紐約
Index:  2 ----> 海灘
Index:  3 ----> 公園
Index:  4 ----> 愛
Index:  5 ----> 快樂
Index:  6 ----> 喜歡
Index:  7 ----> 感謝
Index:  8 ----> 今天
Index:  9 ----> 朋友
```

- LSTM 訓練結果, accuracy為0.9252

```
[ ]                     validation_data=(X_test,  y_test_cat),  callbacks=[early_stop])

Epoch 1/20
80/80 [==============================] - 17s 144ms/step - loss: 2.9916 - accuracy: 0.0646 - val_loss: 2.9696 - val_accuracy: 0.1642
Epoch 2/20
80/80 [==============================] - 5s 62ms/step - loss: 2.6815 - accuracy: 0.1615 - val_loss: 2.2891 - val_accuracy: 0.3217
Epoch 3/20
80/80 [==============================] - 3s 39ms/step - loss: 1.8655 - accuracy: 0.4010 - val_loss: 1.4127 - val_accuracy: 0.6274
Epoch 4/20
80/80 [==============================] - 3s 36ms/step - loss: 1.2201 - accuracy: 0.6393 - val_loss: 0.9182 - val_accuracy: 0.8025
Epoch 5/20
80/80 [==============================] - 2s 28ms/step - loss: 0.8480 - accuracy: 0.7734 - val_loss: 0.6609 - val_accuracy: 0.8574
Epoch 6/20
80/80 [==============================] - 2s 21ms/step - loss: 0.6163 - accuracy: 0.8513 - val_loss: 0.5548 - val_accuracy: 0.8770
Epoch 7/20
80/80 [==============================] - 1s 16ms/step - loss: 0.4681 - accuracy: 0.8949 - val_loss: 0.4835 - val_accuracy: 0.8918
Epoch 8/20
80/80 [==============================] - 1s 15ms/step - loss: 0.3993 - accuracy: 0.9126 - val_loss: 0.4440 - val_accuracy: 0.8977
Epoch 9/20
80/80 [==============================] - 1s 15ms/step - loss: 0.3200 - accuracy: 0.9308 - val_loss: 0.4152 - val_accuracy: 0.9118
Epoch 10/20
80/80 [==============================] - 2s 19ms/step - loss: 0.2714 - accuracy: 0.9499 - val_loss: 0.3806 - val_accuracy: 0.9193
Epoch 11/20
80/80 [==============================] - 1s 18ms/step - loss: 0.2267 - accuracy: 0.9536 - val_loss: 0.3751 - val_accuracy: 0.9224
Epoch 12/20
80/80 [==============================] - 2s 19ms/step - loss: 0.2037 - accuracy: 0.9625 - val_loss: 0.3817 - val_accuracy: 0.9252
Epoch 12: early stopping
<keras.src.callbacks.History at 0x787286ee6fe0>
```

```
[ ]  model.evaluate(X_test,  y_test_cat,  batch_size=batch_size)

20/20 [==============================] - 0s 5ms/step - loss: 0.3817 - accuracy: 0.9252
[0.3817062973976135, 0.9251567125320435]
```

- BLSTM 訓練結果，accuracy為0.9357

```
[ ]  batch_size=64
     epochs  =  15

     model.fit(X_train,  y_train_cat,  batch_size=batch_size,  epochs  =  epochs,
                         shuffle=True,validation_data=(X_test,  y_test_cat),  callbacks=[early_stop])

Epoch 1/15
160/160 [==============================] - 20s 75ms/step - loss: 2.9883 - accuracy: 0.0596 - val_loss: 2.8748 - val_accuracy: 0.1924
Epoch 2/15
160/160 [==============================] - 5s 33ms/step - loss: 2.2178 - accuracy: 0.2835 - val_loss: 1.2829 - val_accuracy: 0.6034
Epoch 3/15
160/160 [==============================] - 3s 19ms/step - loss: 1.0100 - accuracy: 0.6821 - val_loss: 0.5419 - val_accuracy: 0.8417
Epoch 4/15
160/160 [==============================] - 4s 26ms/step - loss: 0.4675 - accuracy: 0.8623 - val_loss: 0.3739 - val_accuracy: 0.8981
Epoch 5/15
160/160 [==============================] - 3s 17ms/step - loss: 0.2848 - accuracy: 0.9240 - val_loss: 0.3265 - val_accuracy: 0.9169
Epoch 6/15
160/160 [==============================] - 4s 26ms/step - loss: 0.1901 - accuracy: 0.9555 - val_loss: 0.3109 - val_accuracy: 0.9310
Epoch 7/15
160/160 [==============================] - 2s 15ms/step - loss: 0.1427 - accuracy: 0.9674 - val_loss: 0.3297 - val_accuracy: 0.9357
Epoch 7: early stopping
<keras.src.callbacks.History at 0x787286c59d80>
```

```
[ ]  model.evaluate(X_test,  y_test_cat,  batch_size=batch_size)

40/40 [==============================] - 0s 9ms/step - loss: 0.3297 - accuracy: 0.9357
[0.329668253660202, 0.9357366561889648]
```

- 20種emojis的數據（precision, recall, f1-score, and total accuracy）

```
80/80 [==============================] - 0s 6ms/step
              precision    recall  f1-score   support

           0       0.62      0.41      0.49       120
           1       0.86      0.77      0.81       107
           2       0.81      0.78      0.79       138
           3       0.93      0.97      0.95       122
           4       0.99      0.98      0.99       121
           5       0.89      0.99      0.94       138
           6       0.91      0.98      0.94       130
           7       0.97      1.00      0.98       129
           8       0.90      1.00      0.95       128
           9       0.91      0.98      0.94       130
          10       0.95      0.99      0.97       124
          11       1.00      0.91      0.95       134
          12       1.00      1.00      1.00       111
          13       0.99      0.97      0.98       139
          14       0.98      1.00      0.99       134
          15       1.00      0.98      0.99       125
          16       0.97      1.00      0.98       123
          17       0.98      1.00      0.99       117
          18       0.98      0.98      0.98       149
          19       0.95      1.00      0.97       133

    accuracy                           0.94      2552
   macro avg       0.93      0.93      0.93      2552
weighted avg       0.93      0.94      0.93      2552
```

- 手動輸入一些測資，跑出的對應表情符號也確實有正確符合語意。

```
Enter tweet
今天太陽真大好熱
 Emojified Tweet
1/1 [==============================] - 0s 20ms/step
今天太陽真大好熱 🌞
```

```
Enter tweet
開始聖誕假期
 Emojified Tweet
1/1 [==============================] - 0s 44ms/step
開始聖誕假期 🎄
```

```
Enter tweet
喜歡你噎
 Emojified Tweet
1/1 [==============================] - 0s 29ms/step
喜歡你噎 💕
```

2. 初步爬蟲測試

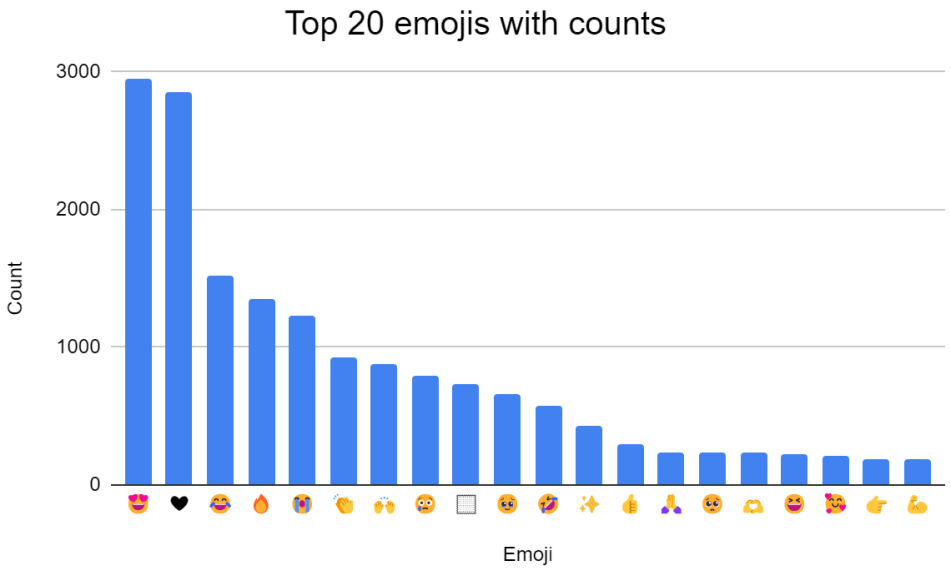
經過翻譯樣本測試後，我們確定中文版本是可執行的。而我們發現翻譯的樣本常會有翻譯錯誤的現象，不符合中文語法，且對應的20種emoji也並不符合我們想要的真實數據，因此我們決定選用更貼近中文使用表情符號狀況的Instagram貼文及留言當作dataset。

利用python套件"instaloader"進行爬蟲，將該帳號前10篇貼文的內容與留言全部擷取。結果可以達到約7成的accuracy，但我們發現選取全部留言的dataset，會使得有太多重複的內容，例如"台灣加油"配上綠色愛心，而這會影響到最終的預測結果。

3. 第一次採用的dataset

根據上面的測試確定爬蟲及其擷取的資料有一定的可行性。我們接著蒐集更多樣化的Instagram帳號資料，並且限制了每次爬取貼文的留言數量，避免過多重複資料。

經過前述所提及的文字預處理及篩選前20名常出現的emoji資料方法，我們得出:



- LSTM，accuracy為0.7028

```
Epoch 6/20
369/369 [==============================] - 51s 139ms/step - loss: 1.2203 - accuracy: 0.6356 - val_loss: 1.2143 - val_accuracy: 0.6400
Epoch 7/20
369/369 [==============================] - 58s 156ms/step - loss: 1.1647 - accuracy: 0.6544 - val_loss: 1.1794 - val_accuracy: 0.6533
Epoch 8/20
369/369 [==============================] - 52s 141ms/step - loss: 1.1099 - accuracy: 0.6696 - val_loss: 1.1538 - val_accuracy: 0.6627
Epoch 9/20
369/369 [==============================] - 54s 147ms/step - loss: 1.0655 - accuracy: 0.6859 - val_loss: 1.1274 - val_accuracy: 0.6698
Epoch 10/20
369/369 [==============================] - 52s 140ms/step - loss: 1.0345 - accuracy: 0.6930 - val_loss: 1.1038 - val_accuracy: 0.6732
Epoch 11/20
369/369 [==============================] - 53s 143ms/step - loss: 1.0015 - accuracy: 0.7005 - val_loss: 1.0919 - val_accuracy: 0.6818
Epoch 12/20
369/369 [==============================] - 50s 136ms/step - loss: 0.9718 - accuracy: 0.7102 - val_loss: 1.0906 - val_accuracy: 0.6879
Epoch 13/20
369/369 [==============================] - 52s 142ms/step - loss: 0.9458 - accuracy: 0.7173 - val_loss: 1.0740 - val_accuracy: 0.6885
Epoch 14/20
369/369 [==============================] - 52s 141ms/step - loss: 0.9288 - accuracy: 0.7223 - val_loss: 1.0675 - val_accuracy: 0.6913
Epoch 15/20
369/369 [==============================] - 51s 139ms/step - loss: 0.9074 - accuracy: 0.7270 - val_loss: 1.0581 - val_accuracy: 0.6949
Epoch 16/20
369/369 [==============================] - 52s 141ms/step - loss: 0.8926 - accuracy: 0.7303 - val_loss: 1.0564 - val_accuracy: 0.6976
Epoch 17/20
369/369 [==============================] - 56s 153ms/step - loss: 0.8734 - accuracy: 0.7344 - val_loss: 1.0392 - val_accuracy: 0.7022
Epoch 18/20
369/369 [==============================] - 52s 140ms/step - loss: 0.8558 - accuracy: 0.7376 - val_loss: 1.0548 - val_accuracy: 0.7028
Epoch 18: early stopping
<keras.src.callbacks.History at 0x7e77c44b2500>
```

```
[ ]  model.evaluate(X_test,  y_test_cat,  batch_size=batch_size)

93/93 [==============================] - 4s 41ms/step - loss: 1.0548 - accuracy: 0.7028
[1.0548404455184937, 0.7027806043624878]
```

- BLSTM, accuracy為0.7263

```
Epoch 1/15
738/738 [==============================] - 256s 331ms/step - loss: 2.4459 - accuracy: 0.2354 - val_loss: 1.7079 - val_accuracy: 0.4735
Epoch 2/15
738/738 [==============================] - 233s 316ms/step - loss: 1.5768 - accuracy: 0.5158 - val_loss: 1.3125 - val_accuracy: 0.5876
Epoch 3/15
738/738 [==============================] - 249s 337ms/step - loss: 1.2705 - accuracy: 0.6128 - val_loss: 1.1635 - val_accuracy: 0.6450
Epoch 4/15
738/738 [==============================] - 237s 321ms/step - loss: 1.1076 - accuracy: 0.6681 - val_loss: 1.0799 - val_accuracy: 0.6741
Epoch 5/15
738/738 [==============================] - 232s 315ms/step - loss: 1.0082 - accuracy: 0.6959 - val_loss: 1.0441 - val_accuracy: 0.6937
Epoch 6/15
738/738 [==============================] - 237s 321ms/step - loss: 0.9263 - accuracy: 0.7192 - val_loss: 1.0113 - val_accuracy: 0.7036
Epoch 7/15
738/738 [==============================] - 236s 320ms/step - loss: 0.8674 - accuracy: 0.7358 - val_loss: 1.0055 - val_accuracy: 0.7074
Epoch 8/15
738/738 [==============================] - 230s 312ms/step - loss: 0.8184 - accuracy: 0.7489 - val_loss: 1.0033 - val_accuracy: 0.7179
Epoch 9/15
738/738 [==============================] - 239s 324ms/step - loss: 0.7847 - accuracy: 0.7580 - val_loss: 0.9794 - val_accuracy: 0.7180
Epoch 10/15
738/738 [==============================] - 238s 322ms/step - loss: 0.7487 - accuracy: 0.7672 - val_loss: 0.9872 - val_accuracy: 0.7263
Epoch 10: early stopping
<keras.src.callbacks.History at 0x7e77cb34e710>
```

```
[ ]  model.evaluate(X_test,  y_test_cat,  batch_size=batch_size)

185/185 [==============================] - 16s 85ms/step - loss: 0.9872 - accuracy: 0.7263
[0.9872250556945801, 0.7263479232788086]
```

- 每個emoji各自的precision、recall和f1-score

```
369/369 [==============================] - 23s 61ms/step
              precision    recall  f1-score   support

           0       0.44      0.17      0.25       605
           1       0.36      0.18      0.24       607
           2       0.80      0.66      0.73       586
           3       0.53      0.52      0.52       576
           4       0.66      0.67      0.67       610
           5       0.59      0.64      0.61       581
           6       0.55      0.60      0.57       569
           7       0.72      0.80      0.76       620
           8       0.71      0.29      0.41       581
           9       0.80      0.72      0.76       586
          10       0.81      0.92      0.86       545
          11       0.94      0.91      0.92       571
          12       0.72      0.78      0.75       606
          13       0.72      0.95      0.82       561
          14       0.82      0.91      0.86       578
          15       0.68      0.91      0.78       613
          16       0.85      0.96      0.90       614
          17       0.84      0.96      0.90       624
          18       0.89      1.00      0.94       561
          19       0.74      0.98      0.84       602

    accuracy                           0.73     11796
   macro avg       0.71      0.73      0.70     11796
weighted avg       0.71      0.73      0.70     11796
```

經過訓練後，我們發現BLSTM會有較高的accuracy，因此在最後版本我們以BLSTM的方法去訓練model。

但我們發現，因為部分帳號經過篩選是否有emoji的資料後，所保留的資料集數較多，導致我們訓練資料的來源會不平均，因此最終我們改以在爬蟲時就篩選出包含emoji的資料，讓每個帳號的資料數量是固定一樣的。(下圖為經過tokenize後idx2words = tokenizer.index_word前20的結果)

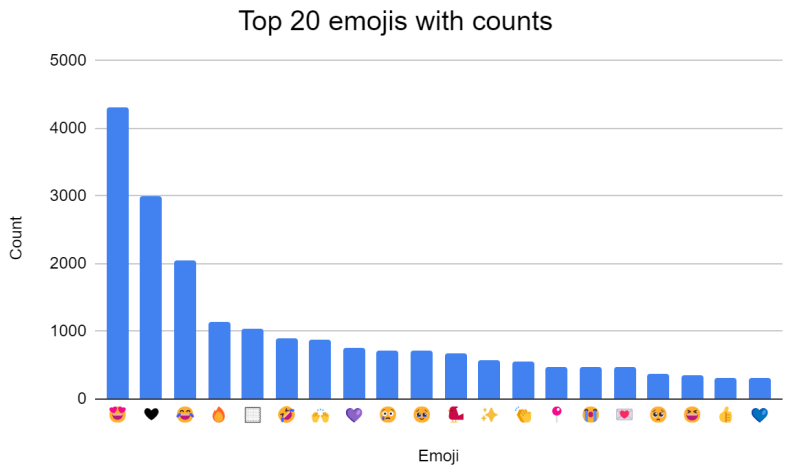
```
Word Mappings

Index:  1 ----> 哥
Index:  2 ----> 真的
Index:  3 ----> 加油
Index:  4 ----> 台灣
Index:  5 ----> 我們
Index:  6 ----> 想
Index:  7 ----> 希望
Index:  8 ----> 總統
Index:  9 ----> 吃
Index:  10 ----> 不
Index:  11 ----> 看
Index:  12 ----> 演唱会
Index:  13 ----> 票
Index:  14 ----> 马来西亚
Index:  15 ----> 抢
Index:  16 ----> 自己
Index:  17 ----> 一起
Index:  18 ----> 加场
Index:  19 ----> 喜歡
```

4. 最後採用的dataset

最後的dataset是固定每個帳號所爬到含有emoji的資料數量。我們分別篩選出前20、30及40個常用的emoji的那些數據資料，並訓練了能夠predict出20、30和40的emoji的model。

a. **top 20 emojis** :

- 出現頻率最高的前20個emojis



- idx2words = tokenizer.index_word 後前20的結果

```
Word Mappings

Index:  1 ----> 想
Index:  2 ----> 一起
Index:  3 ----> 真的
Index:  4 ----> 活動
Index:  5 ----> 喜歡
Index:  6 ----> 參加
Index:  7 ----> 信箱
Index:  8 ----> 發
Index:  9 ----> 友愛
Index:  10 ----> 奶昔
Index:  11 ----> 大哥
Index:  12 ----> 打
Index:  13 ----> 看
Index:  14 ----> 邊吃麥
Index:  15 ----> 當當邊
Index:  16 ----> 影片
Index:  17 ----> 不
Index:  18 ----> 吃
Index:  19 ----> 你們
Index:  20 ----> 自己
```

- 訓練結果, accuracy為0.8039

```
Epoch 1/15
1077/1077 [==============================] - 78s 62ms/step - loss: 2.0724 - accuracy: 0.3556 - val_loss: 1.3255 - val_accuracy: 0.6044
Epoch 2/15
1077/1077 [==============================] - 37s 34ms/step - loss: 1.1836 - accuracy: 0.6516 - val_loss: 0.9406 - val_accuracy: 0.7229
Epoch 3/15
1077/1077 [==============================] - 35s 32ms/step - loss: 0.8989 - accuracy: 0.7399 - val_loss: 0.8083 - val_accuracy: 0.7604
Epoch 4/15
1077/1077 [==============================] - 35s 32ms/step - loss: 0.7628 - accuracy: 0.7793 - val_loss: 0.7470 - val_accuracy: 0.7816
Epoch 5/15
1077/1077 [==============================] - 36s 33ms/step - loss: 0.6833 - accuracy: 0.8020 - val_loss: 0.7185 - val_accuracy: 0.7929
Epoch 6/15
1077/1077 [==============================] - 33s 31ms/step - loss: 0.6303 - accuracy: 0.8159 - val_loss: 0.6848 - val_accuracy: 0.8021
Epoch 7/15
1077/1077 [==============================] - 35s 32ms/step - loss: 0.5918 - accuracy: 0.8256 - val_loss: 0.6866 - val_accuracy: 0.8039
Epoch 7: early stopping
<keras.src.callbacks.History at 0x7d72e039e110>
```

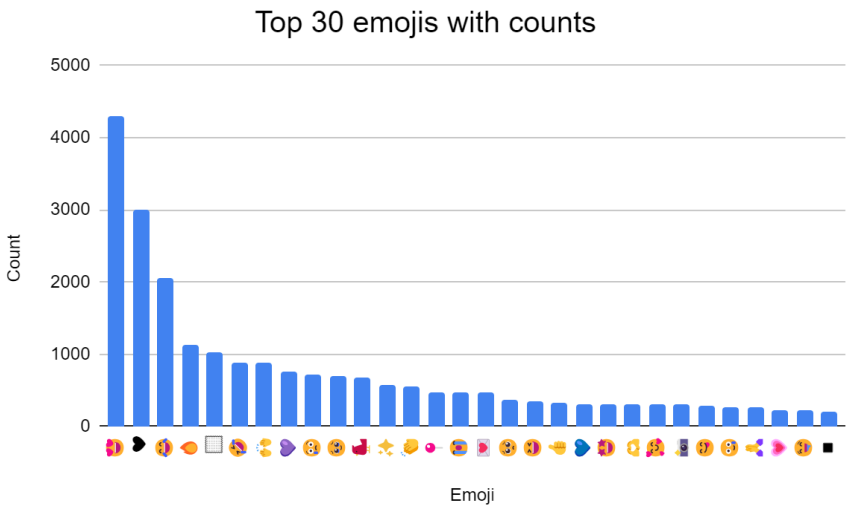
```
[ ]  model.evaluate(X_test,  y_test_cat,  batch_size=batch_size)

270/270 [==============================] - 3s 11ms/step - loss: 0.6866 - accuracy: 0.8039
[0.6865873336791992, 0.8039363622665405]
```

- 每個emoji各自的precision、recall和f1-score

```
539/539 [==============================] - 5s 9ms/step
              precision    recall  f1-score   support

           0       0.46      0.26      0.33       855
           1       0.70      0.34      0.46       884
           2       0.83      0.67      0.74       879
           3       0.65      0.65      0.65       860
           4       0.80      0.58      0.68       893
           5       0.82      0.90      0.86       866
           6       0.67      0.67      0.67       852
           7       1.00      0.96      0.98       871
           8       0.88      0.89      0.89       862
           9       0.80      0.84      0.82       834
          10       0.98      1.00      0.99       853
          11       0.90      0.91      0.90       848
          12       0.74      0.84      0.79       854
          13       1.00      1.00      1.00       850
          14       0.89      0.91      0.90       883
          15       0.93      0.99      0.96       854
          16       0.79      0.93      0.86       875
          17       0.78      0.97      0.86       830
          18       0.71      0.94      0.81       839
          19       0.74      0.99      0.85       882

    accuracy                           0.81     17224
   macro avg       0.80      0.81      0.80     17224
weighted avg       0.80      0.81      0.80     17224
```

b. **top 30 emojis** :

- 出現頻率最高的前30個emojis



- 訓練結果，accuracy為0.8322

```
Epoch 1/15
1615/1615 [==============================] - 86s 48ms/step - loss: 2.1454 - accuracy: 0.3897 - val_loss: 1.2915 - val_accuracy: 0.6400
Epoch 2/15
1615/1615 [==============================] - 49s 30ms/step - loss: 1.2146 - accuracy: 0.6663 - val_loss: 0.9337 - val_accuracy: 0.7371
Epoch 3/15
1615/1615 [==============================] - 47s 29ms/step - loss: 0.9562 - accuracy: 0.7366 - val_loss: 0.8148 - val_accuracy: 0.7760
Epoch 4/15
1615/1615 [==============================] - 47s 29ms/step - loss: 0.8365 - accuracy: 0.7685 - val_loss: 0.7459 - val_accuracy: 0.7919
Epoch 5/15
1615/1615 [==============================] - 48s 30ms/step - loss: 0.7609 - accuracy: 0.7892 - val_loss: 0.7086 - val_accuracy: 0.8012
Epoch 6/15
1615/1615 [==============================] - 47s 29ms/step - loss: 0.7076 - accuracy: 0.8011 - val_loss: 0.6817 - val_accuracy: 0.8076
Epoch 7/15
1615/1615 [==============================] - 47s 29ms/step - loss: 0.6713 - accuracy: 0.8099 - val_loss: 0.6531 - val_accuracy: 0.8156
Epoch 8/15
1615/1615 [==============================] - 47s 29ms/step - loss: 0.6372 - accuracy: 0.8174 - val_loss: 0.6494 - val_accuracy: 0.8204
Epoch 9/15
1615/1615 [==============================] - 46s 29ms/step - loss: 0.6165 - accuracy: 0.8221 - val_loss: 0.6374 - val_accuracy: 0.8195
Epoch 10/15
1615/1615 [==============================] - 48s 29ms/step - loss: 0.5948 - accuracy: 0.8271 - val_loss: 0.6119 - val_accuracy: 0.8263
Epoch 11/15
1615/1615 [==============================] - 47s 29ms/step - loss: 0.5777 - accuracy: 0.8307 - val_loss: 0.6033 - val_accuracy: 0.8278
Epoch 12/15
1615/1615 [==============================] - 46s 29ms/step - loss: 0.5630 - accuracy: 0.8340 - val_loss: 0.5904 - val_accuracy: 0.8298
Epoch 13/15
1615/1615 [==============================] - 47s 29ms/step - loss: 0.5506 - accuracy: 0.8366 - val_loss: 0.6005 - val_accuracy: 0.8322
Epoch 13: early stopping
<keras.src.callbacks.History at 0x7cf3dd36ac80>
```

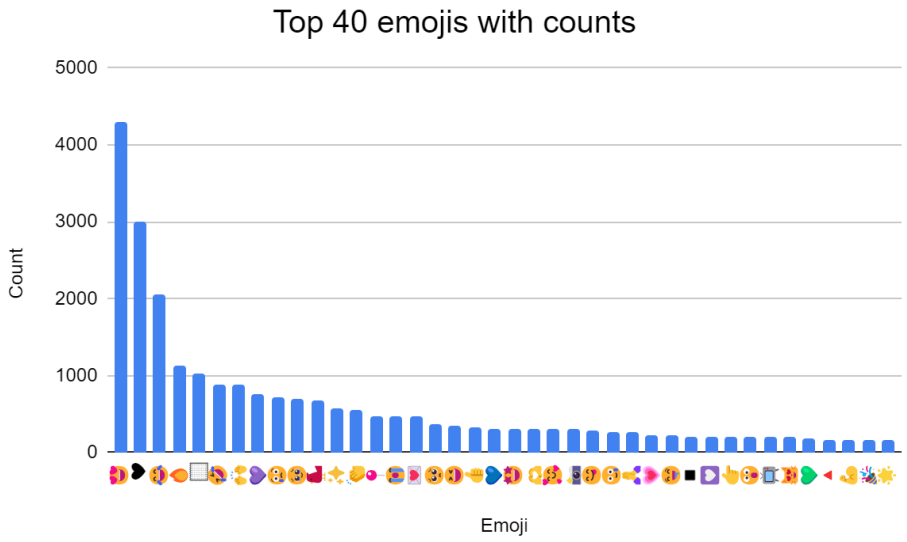
```
[ ]  model.evaluate(X_test,  y_test_cat,  batch_size=batch_size)

     404/404 [==============================] - 8s 19ms/step - loss: 0.6005 - accuracy: 0.8322
     [0.6004922986030579, 0.832172155380249]
```

- 每個emoji各自的precision、recall和f1-score

```
[ ]  808/808 [==============================] - 7s 9ms/step
                precision    recall  f1-score   support

             0       0.53      0.21      0.30       822
             1       0.66      0.32      0.43       878
             2       0.86      0.67      0.75       830
             3       0.71      0.64      0.67       844
             4       0.81      0.44      0.57       838
             5       0.85      0.88      0.87       844
             6       0.67      0.70      0.69       826
             7       0.99      0.97      0.98       901
             8       0.90      0.88      0.89       873
             9       0.82      0.81      0.81       859
            10       0.99      1.00      0.99       869
            11       0.90      0.87      0.88       879
            12       0.86      0.78      0.82       838
            13       0.99      1.00      0.99       831
            14       0.87      0.91      0.89       884
            15       0.91      1.00      0.95       898
            16       0.79      0.85      0.82       908
            17       0.89      0.90      0.89       842
            18       0.75      0.88      0.81       903
            19       0.71      0.98      0.82       851
            20       0.82      0.82      0.82       847
            21       0.68      0.88      0.77       859
            22       0.78      0.89      0.83       859
            23       0.97      0.99      0.98       821
            24       0.78      0.87      0.82       863
            25       0.85      0.99      0.91       860
            26       0.78      0.97      0.86       876
            27       0.89      0.90      0.89       881
            28       0.78      0.92      0.84       877
            29       1.00      1.00      1.00       875

      accuracy                           0.83     25836
     macro avg       0.83      0.83      0.82     25836
  weighted avg       0.83      0.83      0.82     25836
```

c. **top 40 emojis**：

- 出現頻率最高的前40個emojis



- 訓練結果，accuracy為0.8438

```
Epoch 1/15
2153/2153 [==============================] - 103s 43ms/step - loss: 2.1042 - accuracy: 0.4312 - val_loss: 1.2105 - val_accuracy: 0.6763
Epoch 2/15
2153/2153 [==============================] - 69s 32ms/step - loss: 1.1587 - accuracy: 0.6962 - val_loss: 0.8914 - val_accuracy: 0.7628
Epoch 3/15
2153/2153 [==============================] - 68s 32ms/step - loss: 0.9263 - accuracy: 0.7591 - val_loss: 0.7801 - val_accuracy: 0.7929
Epoch 4/15
2153/2153 [==============================] - 68s 31ms/step - loss: 0.8143 - accuracy: 0.7875 - val_loss: 0.7144 - val_accuracy: 0.8105
Epoch 5/15
2153/2153 [==============================] - 68s 32ms/step - loss: 0.7462 - accuracy: 0.8037 - val_loss: 0.6845 - val_accuracy: 0.8166
Epoch 6/15
2153/2153 [==============================] - 67s 31ms/step - loss: 0.6945 - accuracy: 0.8159 - val_loss: 0.6474 - val_accuracy: 0.8247
Epoch 7/15
2153/2153 [==============================] - 67s 31ms/step - loss: 0.6606 - accuracy: 0.8232 - val_loss: 0.6347 - val_accuracy: 0.8290
Epoch 8/15
2153/2153 [==============================] - 64s 30ms/step - loss: 0.6391 - accuracy: 0.8273 - val_loss: 0.6160 - val_accuracy: 0.8324
Epoch 9/15
2153/2153 [==============================] - 71s 33ms/step - loss: 0.6057 - accuracy: 0.8341 - val_loss: 0.6137 - val_accuracy: 0.8342
Epoch 10/15
2153/2153 [==============================] - 67s 31ms/step - loss: 0.5871 - accuracy: 0.8375 - val_loss: 0.5941 - val_accuracy: 0.8369
Epoch 11/15
2153/2153 [==============================] - 67s 31ms/step - loss: 0.5741 - accuracy: 0.8404 - val_loss: 0.5816 - val_accuracy: 0.8399
Epoch 12/15
2153/2153 [==============================] - 70s 33ms/step - loss: 0.5588 - accuracy: 0.8428 - val_loss: 0.5685 - val_accuracy: 0.8424
Epoch 13/15
2153/2153 [==============================] - 67s 31ms/step - loss: 0.5451 - accuracy: 0.8453 - val_loss: 0.5738 - val_accuracy: 0.8438
Epoch 13: early stopping
<keras.src.callbacks.History at 0x7fb4c88f62c0>
```

```
model.evaluate(X_test,  y_test_cat,  batch_size=batch_size)
```

```
539/539 [==============================] - 6s 11ms/step - loss: 0.5738 - accuracy: 0.8438
[0.5738038420677185, 0.8437645435333252]
```

- 每個emoji各自的precision、recall和f1-score

```
1077/1077 [==============================] - 12s 11ms/ste
              precision    recall  f1-score   support

           0       0.59      0.23      0.33       892
           1       0.75      0.27      0.40       843
           2       0.85      0.66      0.74       869
           3       0.68      0.59      0.63       857
           4       0.87      0.33      0.48       857
           5       0.86      0.86      0.86       820
           6       0.71      0.65      0.68       840
           7       0.99      0.96      0.98       903
           8       0.86      0.89      0.87       882
           9       0.80      0.75      0.77       848
          10       0.99      0.99      0.99       843
          11       0.96      0.82      0.88       864
          12       0.77      0.74      0.75       906
          13       0.99      0.98      0.99       845
          14       0.89      0.86      0.87       850
          15       0.91      0.99      0.95       867
          16       0.80      0.79      0.80       811
          17       0.71      0.90      0.80       874
          18       0.74      0.87      0.80       885
          19       0.74      0.95      0.83       890
          20       0.85      0.81      0.83       837
          21       0.70      0.85      0.77       841
          22       0.80      0.84      0.82       836
          23       0.96      1.00      0.98       868
          24       0.85      0.80      0.83       869
          25       0.82      0.98      0.89       800
          26       0.83      0.93      0.87       889
          27       0.88      0.90      0.89       874
          28       0.71      0.97      0.82       857
          29       0.96      1.00      0.98       849
          30       0.96      0.98      0.97       865
          31       0.93      0.83      0.88       882
          32       0.90      0.98      0.94       818
          33       0.91      1.00      0.95       839
          34       0.77      0.92      0.84       872
          35       0.85      0.94      0.90       868
          36       0.99      0.99      0.99       914
          37       0.80      0.99      0.89       852
          38       0.84      1.00      0.91       874
          39       0.96      0.96      0.96       898

    accuracy                           0.84     34448
   macro avg       0.84      0.84      0.83     34448
weighted avg       0.84      0.84      0.83     34448
```

Analysis

5. 測試結果
    a. 第二次dataset版本
    - 自己人工測試的截圖，都有滿符合預期的結果

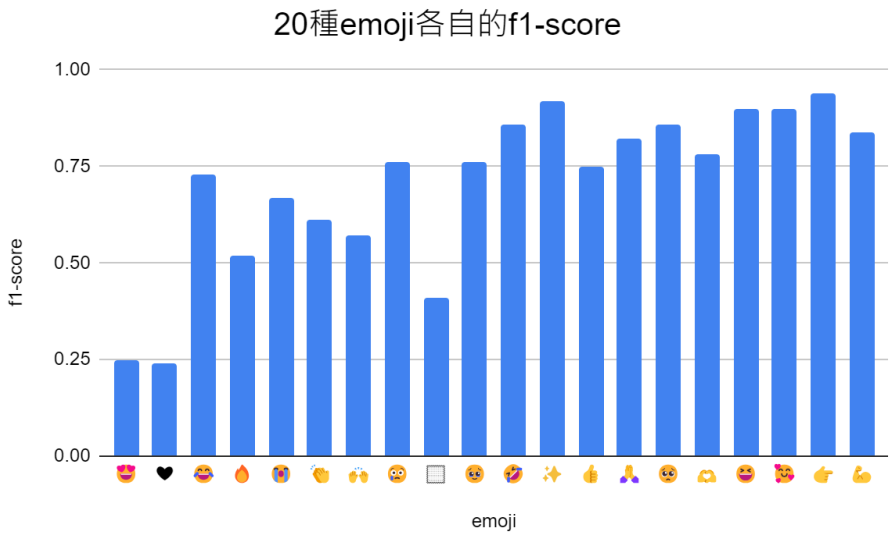
```
Enter text

我真的好想哭


Emojified Text


1/1 [==============================] - 0s 31ms/step
我真的好想哭 😭
```

```
Enter text
希望你一切順利

Emojified Text

1/1 [==============================] - 0s 22ms/step
希望你一切順利 🙏
```

```
Enter text
我上課不專心 不在意 是因為偷偷看你

Emojified Text

1/1 [==============================] - 0s 32ms/step
我上課不專心 不在意 是因為偷偷看你 😍
```

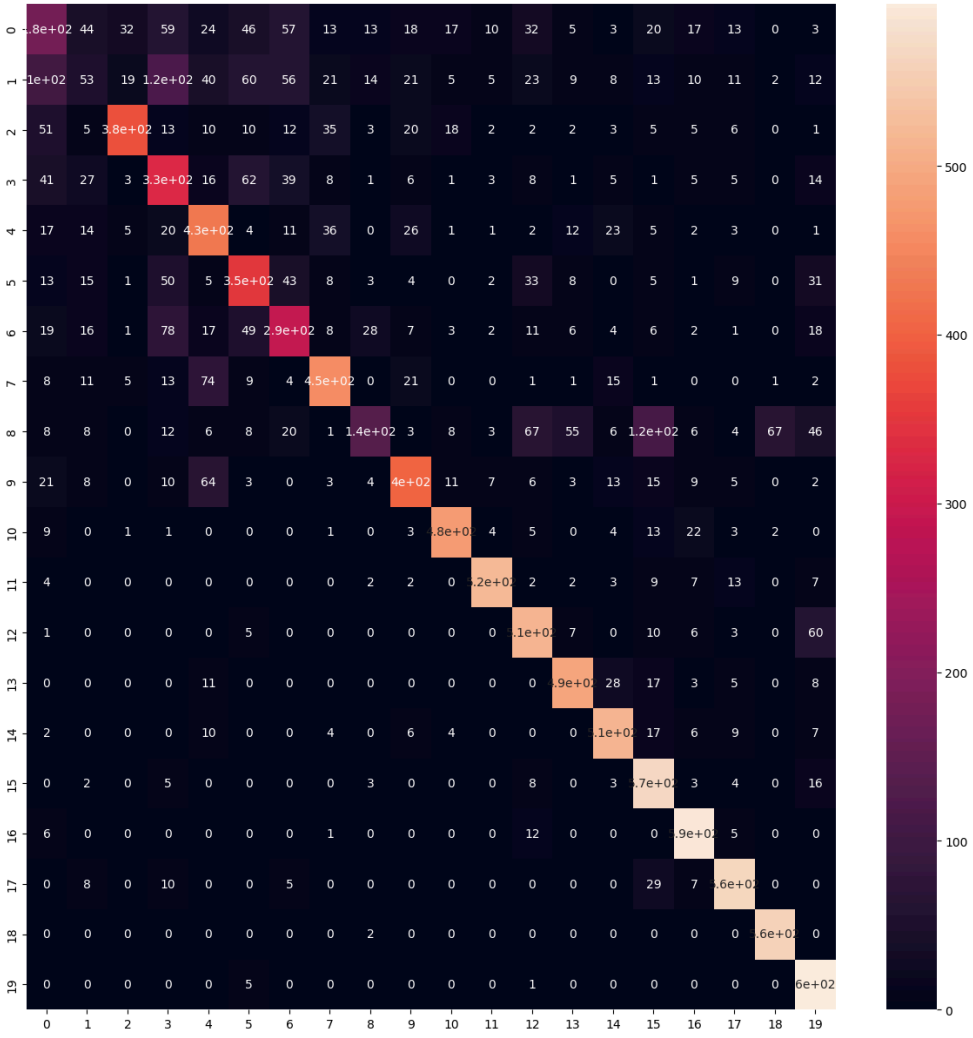
```
Enter text
加油啊

Emojified Text

1/1 [==============================] - 0s 20ms/step
加油啊 💪
```

- 各個emoji的f1-score



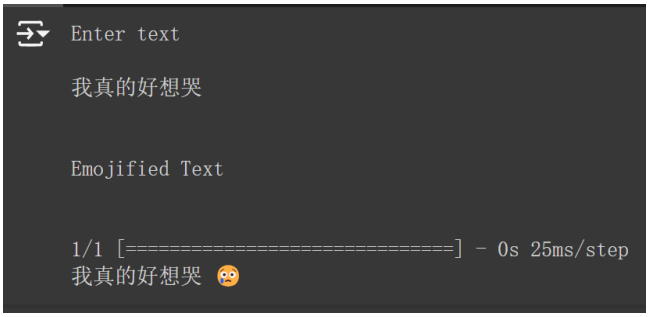
- confusion matrix

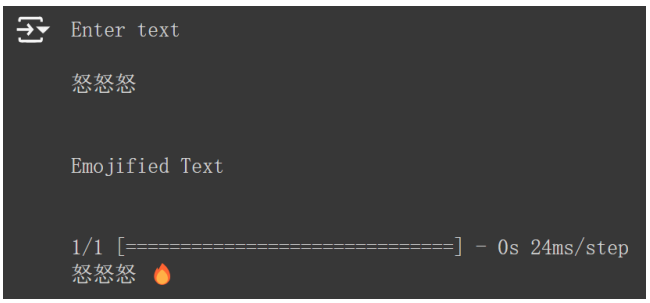
  本研究的confusion matrix和一般較為不同，因為最後分類出20個類別，因此matrix會是20*20，而對角線即是代表TP值，由下圖可以看出TP值都明顯較高。

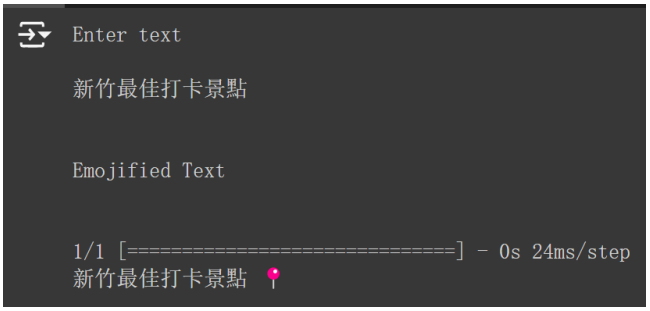


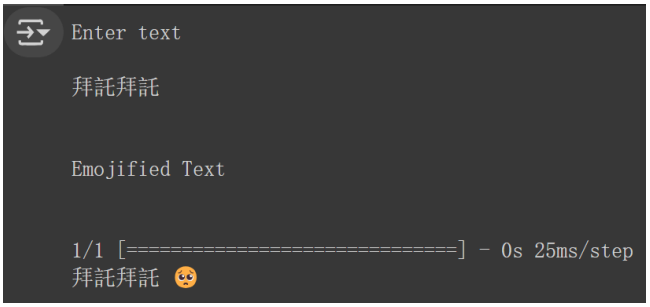
b. 最終採用dataset: top 20 emoji

- 人工測試截圖



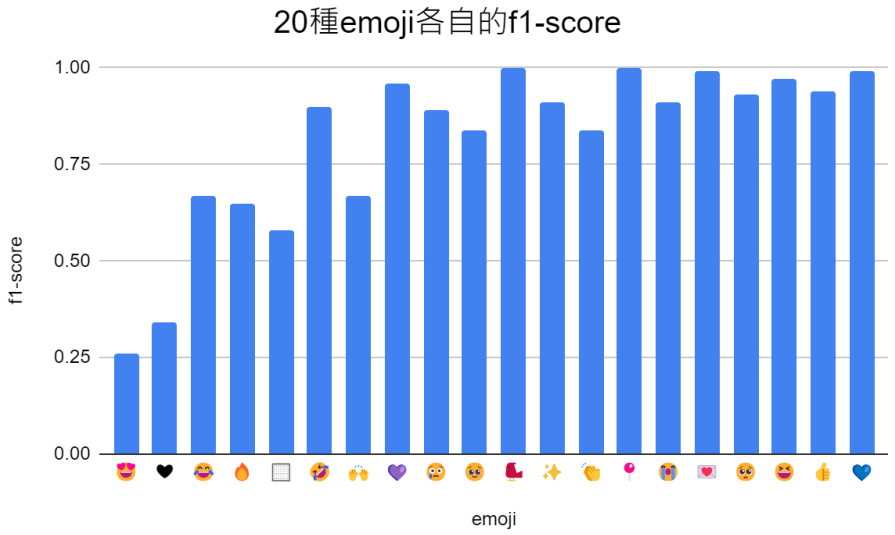




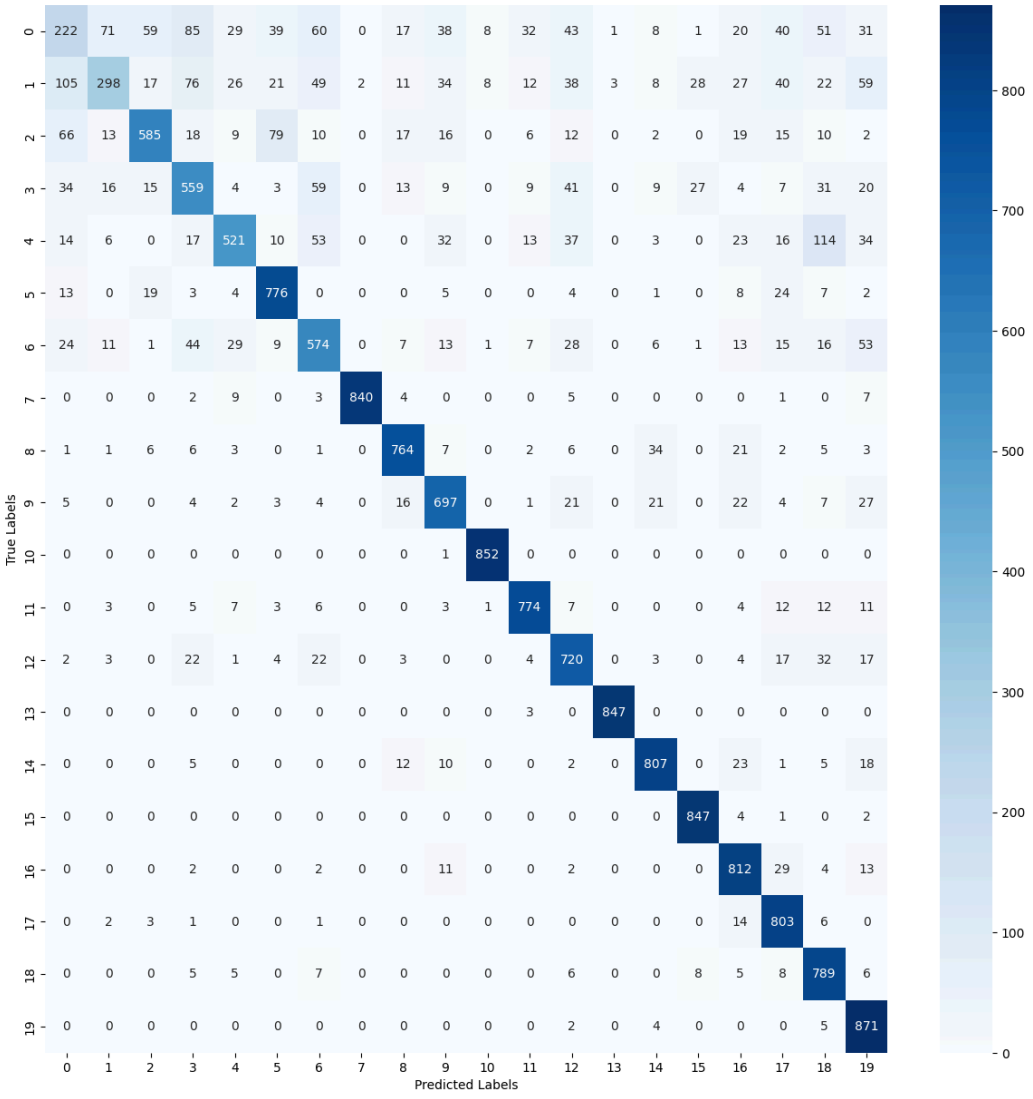


- 各個emoji個別的分析結果

  可以看到前面幾名，代表比較常出現的emoji，其f1-score較低。我們推測因資料集訓練前會有over-sampling，使得排名靠後的emoji資料被大量複製，而排名靠前的資料重複性就會較低。這可能讓模型在高重複性的feature上學習得更好，造成出現頻率低的emoji準確率更高，f1-score表現較前幾名佳。

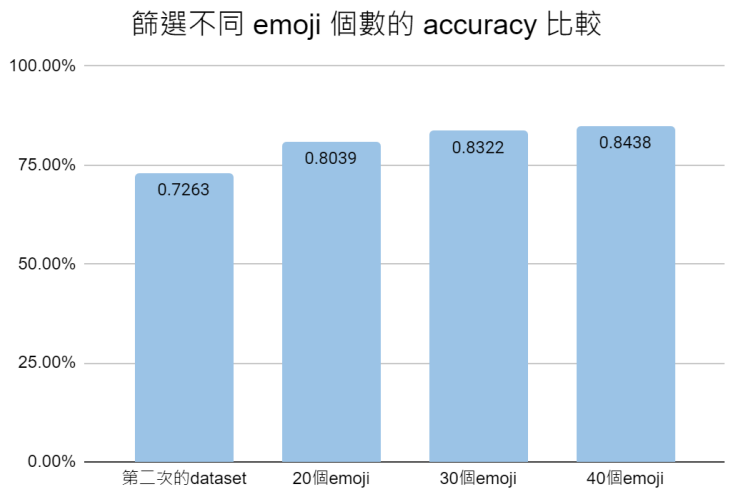


- confusion matrix（同前述的confusion matrix分析，而可以看到較前面的label，即為較常出現的emoji，其TP值都會較低。）



6. 不同model的accuracy比較

   原先第二次的dataset accuracy約為70%，而最終dataset的版本則提升到了80%，且篩選的emoji個數越多，其accuracy有逐漸增加的趨勢。



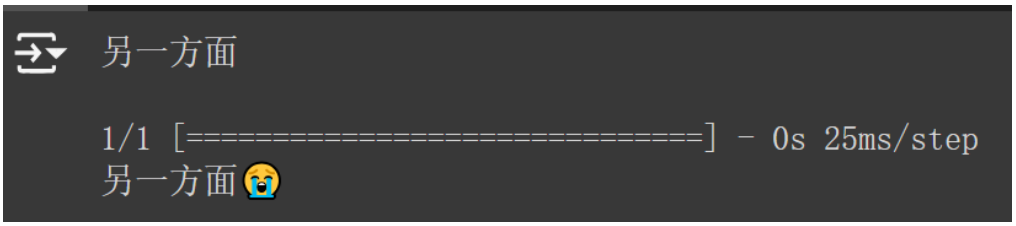
7. 結論
   - 經過多次測試和資料集調整，我們發現BLSTM模型對於多種類emoji預測效果最佳。
   - 使用更貼近中文實際使用情境的Instagram資料集，並進行適當的資料篩選和預處理以維持多元性，提升模型準確率。
   - 最終模型的accuracy較baseline低，可能的原因是dataset多元性問題，如，1)我們爬的資料會受限於Twitter英文用戶與instagram中文用戶數量差異導致的留言多元性問題2)Twitter的資料是從最近七日所有推文爬取，而Instagram要限定帳號爬取資料，降低了多元性。
   - 所有related works都只有20種emojis，而我們的最終模型能夠有效預測多達40種常用emoji。

Error Analysis

我們的model在一些特定輸入中可能存在問題:

1. 輸入空值
   不會產生任何emoji，output為僅含空格的字串。
2. 輸入只含有stopwords的句子
   input:雖然

input:另一方面

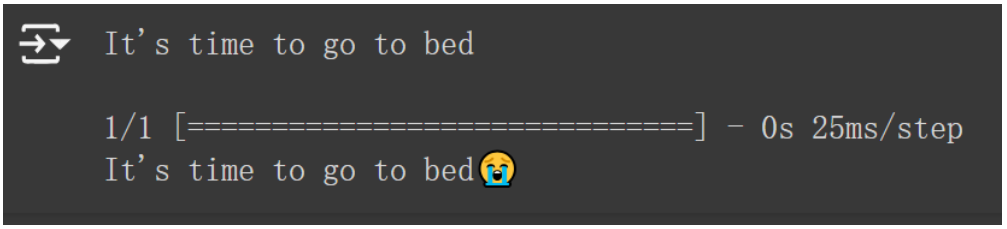


input:但是那時候

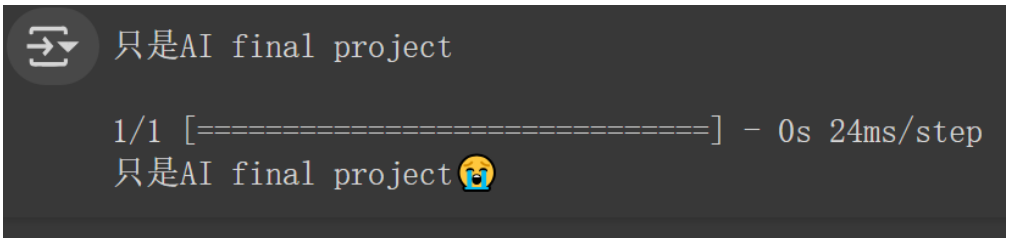
如上方範例所示，所有只包含stopwords的句子，資料預處理之後將會是空值，輸出結果皆會相同。

3. 輸入非中文

input:It's time to go to bed



input:只是AI final project

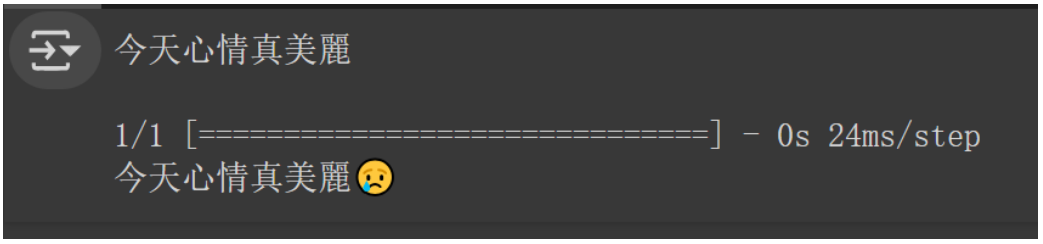


input:喜歡你yeah

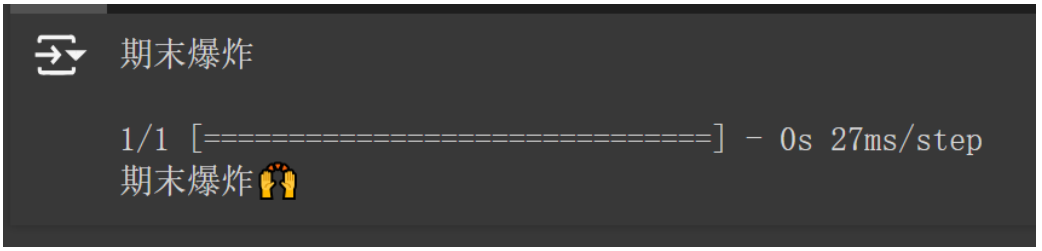
如上所示，當input只有非中文和stopwords時，輸出結果將如同只有stopwords一樣，會輸出固定的emoji。若input含有非中文及非stopwords，輸出結果將依據非stopwords的部分產生。

4. 不合理的輸出

input:今天心情真美麗

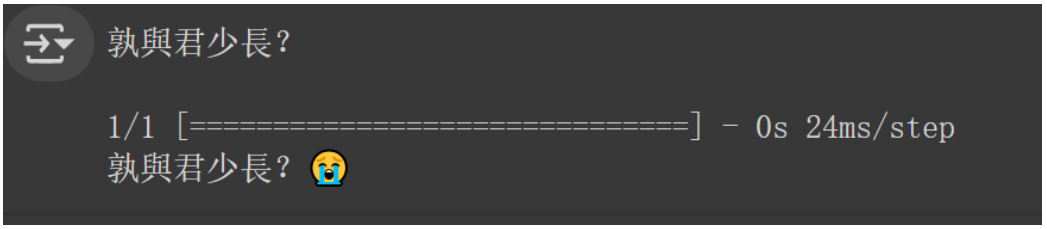


input:期末爆炸



input:這真的超爛

input:孰與君少長？



如上所示，由於dataset中負面詞語較少，因此在遇到帶有負面情緒輸入時，模型會產出不合理的輸出。由於我們採集的資料內容皆為日常用語，因此輸入文言文，也就是模型未學習過的詞語時，會產生不合理的結果。

Future Work

1. 更大、更多樣化的資料集:擴展資料集以包含更多樣化的中文文字來源可以提高模型在不同情境和溝通方式中的泛化能力。
2. 預處理技術:實作更複雜的文字預處理技術，例如處理俚語、縮寫和特定於上下文的短語，可以幫助模型更好地理解語言的細微差別。
3. 增加emoji類別:增加模型可預測的表情符號類別，以涵蓋更多的情感和表達
4. 一句話有多個emoji:處理一句話中出現多個emoji的情況。目前的模型只專注於一個emoji，但在現實中，句子中經常會出現多個emoji來表達複雜的情感。
5. 即時回饋:整合即時回饋循環，使用者可以在其中提供有關建議表情符號的回饋，有助於增加模型的多元性。

挑戰與解決方案

1. 挑戰:處理文字到表情符號翻譯中固有的歧義性和主觀性可能很困難，因為相同的文字可能會引起不同使用者的不同情緒。

    a. 解決方案：將使用者特定的偏好和歷史資料納入模型中可以幫助根據個人使用者的風格和偏好自訂預測。
2. 挑戰：模型可能會遇到不太常見或新的表情符號，這些表情符號在訓練資料中沒有很好的體現。
    a. 解決方案：定期更新訓練資料集以包含最新和趨勢的表情符號可以保持模型的相關性和準確性。
3. 挑戰：確保模型的建議在不同文化背景下適合情境可能很複雜。
    a. 解決方案：擴大dataset的使用情境及面向，幫助完善模型對表情符號使用中文化細微差別的理解。

Code

https://github.com/hyching21/Intro-to-AI-Final-Project-Team12

Contribution of each member

name	contribution(%)
蔡芳慈 111550024	25
陳奕 111550168	25
謝詠晴 111550113	25
郭芷杆 111550155	25

References

- **Andy Lolu (2021)**. Text2Emoji. https://github.com/andylolu2/Text2Emoji.
- **Aguschin (2020)**. Lyrics to Emoji. https://www.kaggle.com/code/aguschin/lyrics-to-emoji.
- **Defcon27 (2021)**. Emoji Prediction using Deep Learning. https://github.com/Defcon27/Emoji-Prediction-using-Deep-Learning/tree/master.
- **Leya O Liat & Arun Mohan (2022)**. Sentiments in Tweets with Emojis. https://github.com/leyaoliatan/Sentiments-in-Tweets-with-Emojis/tree/main.
- **WCSE (2020)**. Exploring the Impact of Emoji in Sentiment Analysis. https://www.wcse.org/WCSE_2020_Summer/011.pdf.
- **Hugging Face (2020)**. Tweet Eval: A Unified Benchmark and Comparative Evaluation for Tweet Classification. https://huggingface.co/datasets/tweet_eval.